体系结构第二次实验

PB18000221 袁一玮

## 实验目标

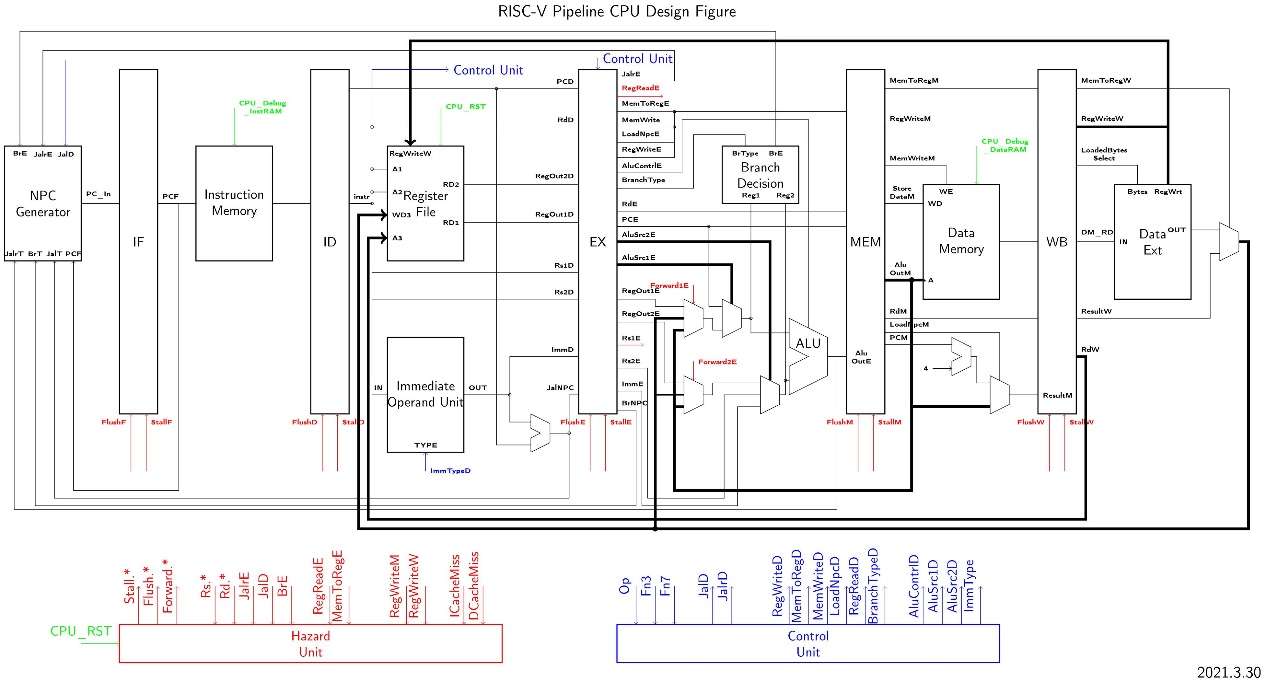
利用助教给出的实验框架，补全流水线的设计，同时加入CSR的寄存器及走线

## 实验环境和工具

Vivado 2019，Windows 10，VS Code

## 阶段1&2

### 设计图



### 实验内容

在助教的框架上，填补了ALU、BranchDecisionMaking、ControlUnit、DataExt、HarzardUnit、ImmOperandUnit等模块，其中ControlUnit和HarzardUnit较为复杂

在ControlUnit中，根据Op和对应的Fn3、Fn7来给出AluContrlD、RegReadD等信号，基本上就是照着助教给出的指令格式写case。

在HarzardUnit中，要检测指令是否会把目标寄存器写入的新数据转发至EX段进入ALU，也要检测是否会出现转发无法解决的数据依赖：

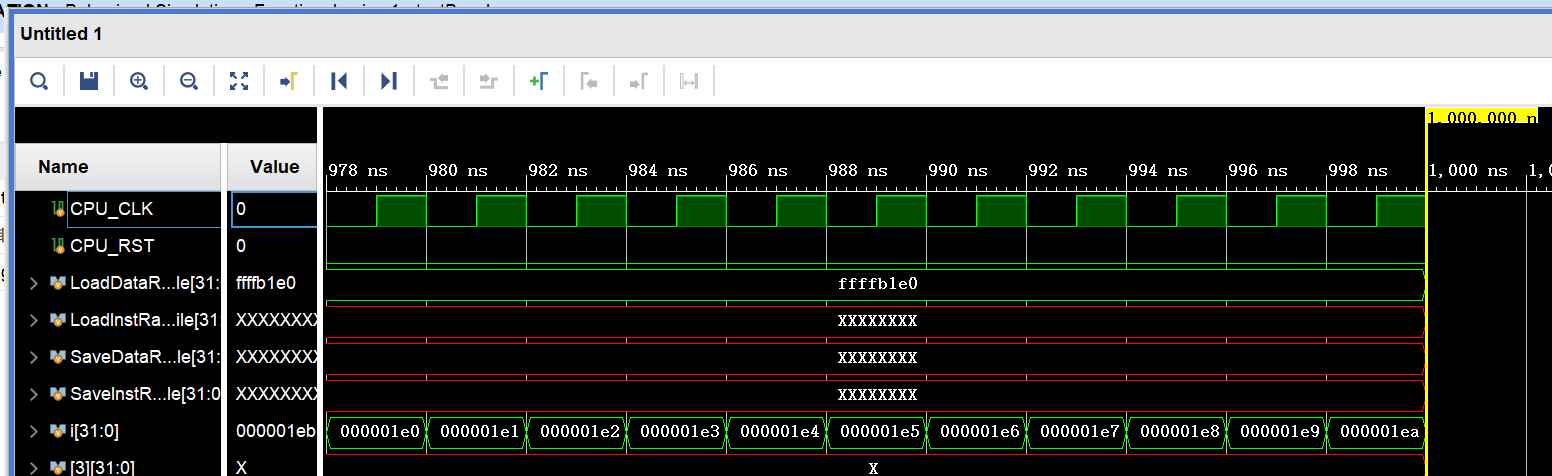
若需要Mem段数据转发至ALU，即有ForwardE <= 2'b10；若是WB阶段的数据转发至ALU，即有ForwardE <= 2'b01；若无需转发，则有ForwardE <= 2'b00，保持原来流水线数据通路。

当指令类型为跳转指令时，经历了IF -> ID -> JUMP，即需要{FlushD,FlushE} <= 2'b11来清空ID、EX阶段的寄存器数据，从新的指令开始（清空流水线的控制冒险）。

出现从Mem写回Reg且数据依赖的情况时（暂停一个时钟周期的数据依赖冒险），需要stall即{StallF,StallD,FlushE} <= 3'b111，来更新Reg的内容。

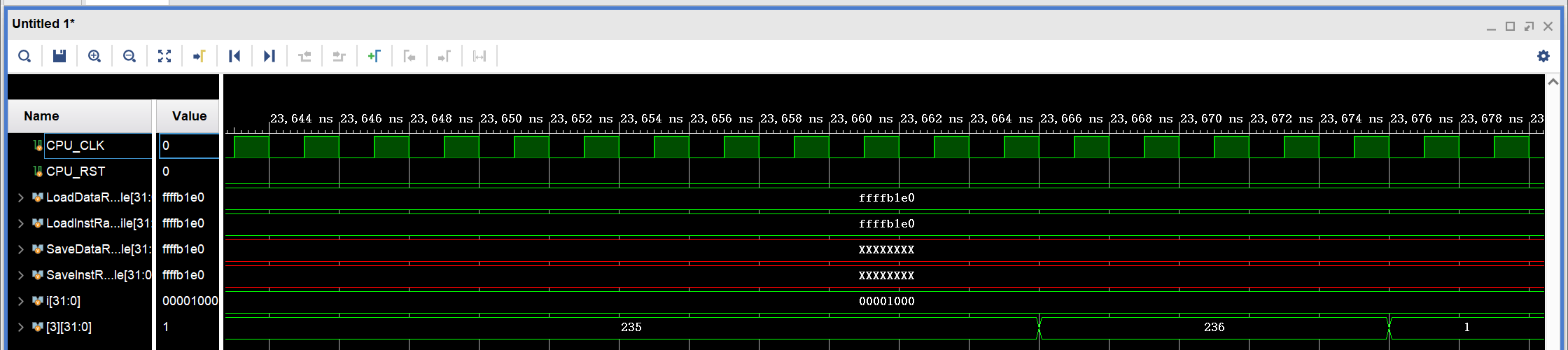
### 实验结果

默认情况下，仿真只会进行到1000ns；我们需要用Run All的仿真按钮来运行直到$stop



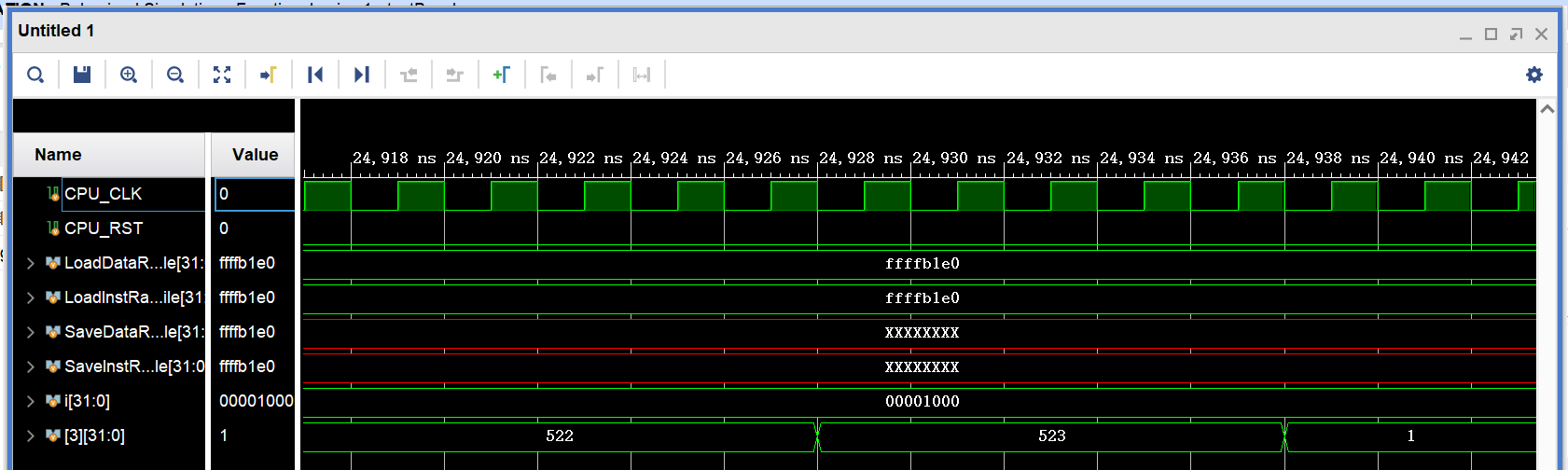
对于1testAll：

在gp第三个寄存器从236变成1，通过测试



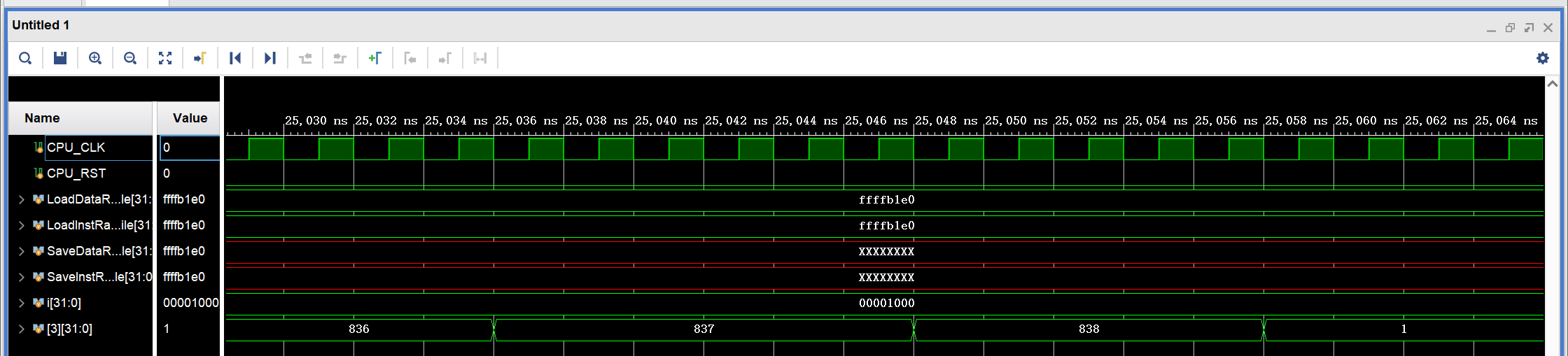
对于2testAll：

在gp从523变成1，通过测试



对于3testAll：

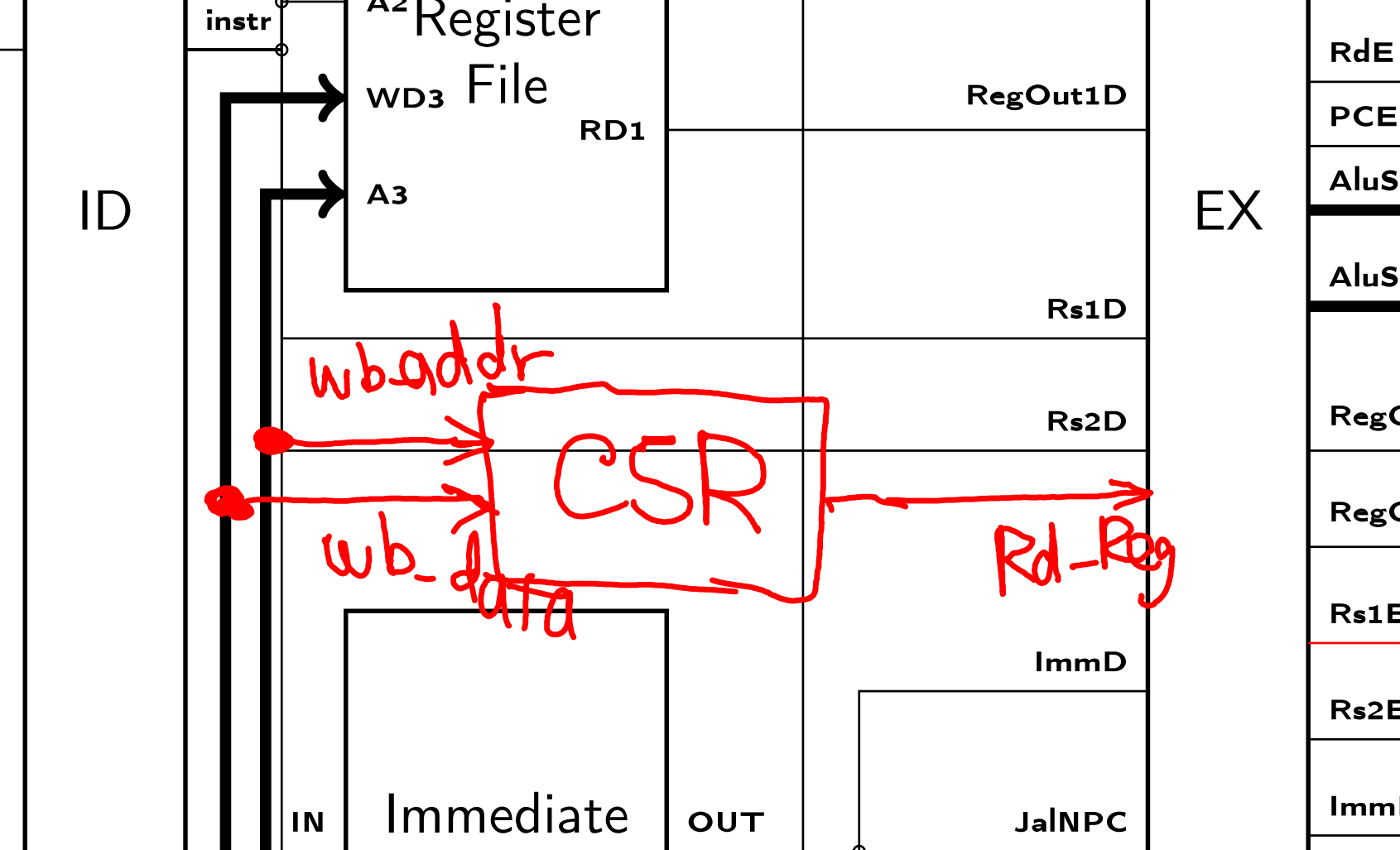
在gp从838变成1，通过测试



## 阶段3

### 设计图

在寄存器文件附近的ID/EX加入CSR



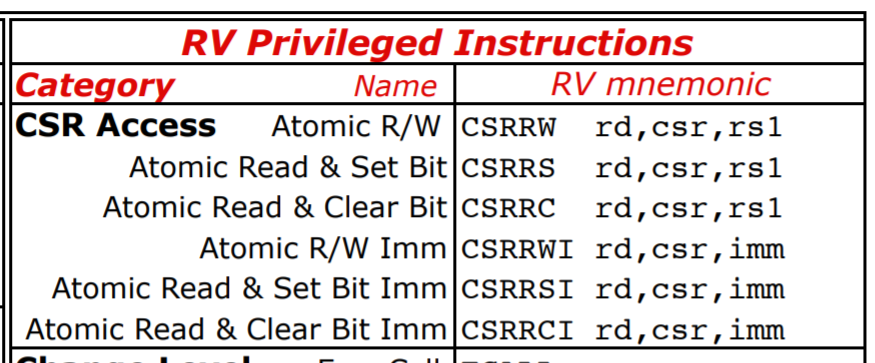
### 实验内容

CSR寄存器使用专有的12位地址编码空间，在顶层设计中加入了CSR的数据线和地址线：

EX阶段需要像通用寄存器一样引入CSR的数据线和地址线，同时在ALU中加入取反与操作和直接读取寄存器所需要的指令（本来LUI只能直接读取ALU输入2，现在也需要直接读取ALU输入1的指令）

同时在HarzardUnit中，也需要和ALU的两个输入一样，对寄存器进行转发：

Mem中数据将在WB阶段写回CSR中：若需要在EX阶段使用Mem段数据，即转发至ALU，即有ForwardE <= 2'b10；若需要在EX阶段使用WB段数据，即有ForwardE <= 2'b01；若无需转发，则有ForwardE <= 2'b00，保持原来流水线数据通路



CSR寄存器，和通用寄存器类似：

module **CSR**(

    input wire clk,

    input wire rst,

    input wire write\_en,

    input wire [11:0] addr, wb\_addr,

    input wire [31:0] wb\_data,

    output wire [31:0] rd\_reg

    );

    reg [31:0] reg\_file[31:0];

    integer i;

    initial begin

        for(i = 0; i < 32; i = i + 1)

            reg\_file[i][31:0] <= 32'b1;

    end

    always@(negedge clk or posedge rst) begin

        if (rst)

            for (i = 0; i < 32; i = i + 1)

                reg\_file[i][31:0] <= 0;

        else if(write\_en)

            reg\_file[wb\_addr] <= wb\_data;

    end

    assign rd\_reg = reg\_file[addr];

endmodule

### 实验结果

使用了去年学长给出的测试文件

csrtest.om:     file format elf32-littleriscv

Disassembly of section .text:

00010054 <test\_0>:

   10054:   00000193             li gp,0

   10058:   00f00093             li ra,15

   1005c:   00009073             csrw  ustatus,ra

   10060:   00003173             csrrc sp,ustatus,zero

   10064:   06111063             bne   sp,ra,100c4 <failed>

   10068:   000c7073             csrci ustatus,24

   1006c:   00003173             csrrc sp,ustatus,zero

   10070:   00700093             li ra,7

   10074:   04111863             bne   sp,ra,100c4 <failed>

00010078 <test\_2>:

   10078:   00200193             li gp,2

   1007c:   00100093             li ra,1

   10080:   00209073             fsrm  ra

   10084:   002c6173             csrrsi   sp,frm,24

   10088:   02111e63             bne   sp,ra,100c4 <failed>

   1008c:   00201173             fsrm  sp,zero

   10090:   01900093             li ra,25

   10094:   02111863             bne   sp,ra,100c4 <failed>

00010098 <test\_3>:

   10098:   00300193             li gp,3

   1009c:   003c5073             csrwi fcsr,24

   100a0:   00700093             li ra,7

   100a4:   0030a173             csrrs sp,fcsr,ra

   100a8:   01800093             li ra,24

   100ac:   00111c63             bne   sp,ra,100c4 <failed>

   100b0:   00301173             fssr  sp,zero

   100b4:   01f00093             li ra,31

   100b8:   00111663             bne   sp,ra,100c4 <failed>

000100bc <success>:

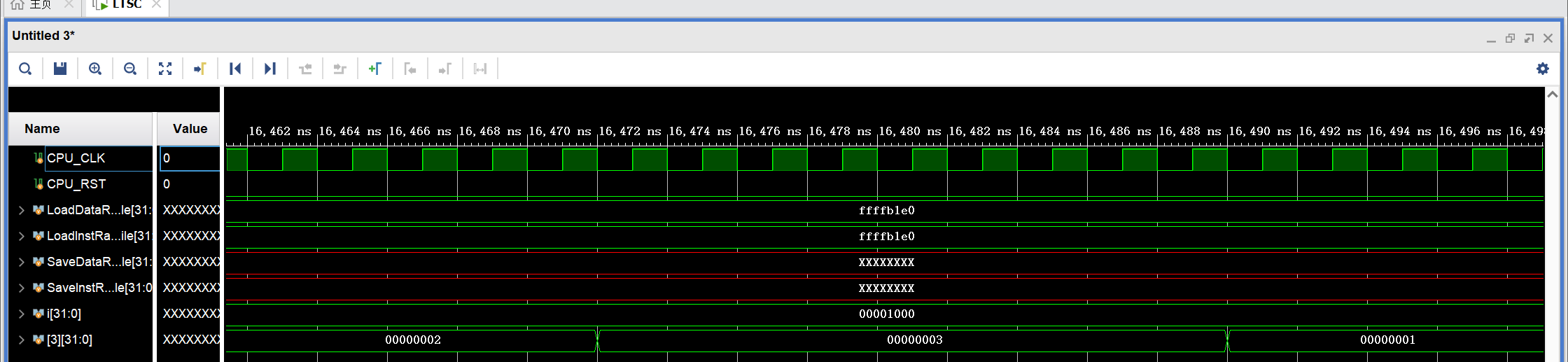
   100bc:   00100193             li gp,1

   100c0:   ffdff06f             j  100bc <success>

000100c4 <failed>:

   100c4:   0000006f             j  100c4 <failed>

可以看到仿真结果里三号寄存器从3变成1，测试通过



## 实验总结和想法

助教让我们用三个星期基于给出的框架完成流水线是不难的，但是因为假期，所以最后只用了两三天草草完成了流水线

在加CSR数据通路的时候，要对顶层部分及ALU、控制单元进行额外的修改，但是助教在这些文件头部备注中的无须修改让人有点迷惑，可以在这个时候表明是哪个阶段无须修改、哪些阶段是有必要改数据通路的